Network representations: Semantic Networks. Frames

What is knowledge representation?

What is representation?

- Representation refers to a symbol or thing which represents ('refers to', 'stands for') something else.

When do we need to represent?

- We need to represent a thing in the natural world when we don't have, for some reason, the possibility to use the original 'thing'.
- Example: Planning ahead – how will our actions affect the world, and how will we reach our goals?

The object of knowledge representation is to express the problem in computer-understandable form

Aspects of KR

Syntactic
- Possible (allowed) constructions
- Each individual representation is often called a **sentence**.
- For example: *color(my_car, red), my_car(red), red(my_car), etc.*

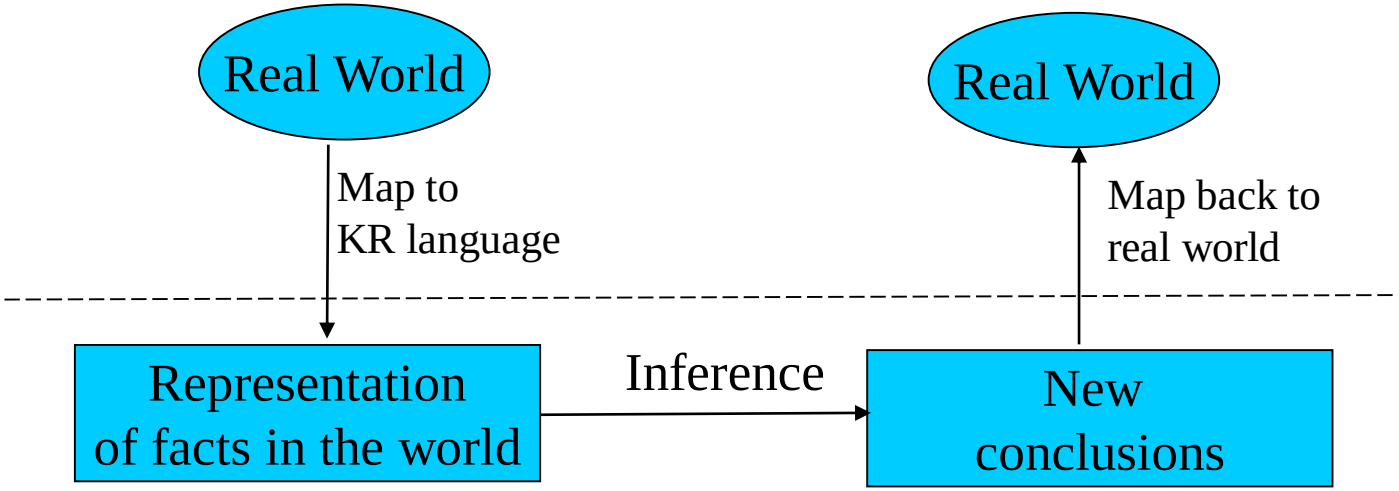
Semantic
- What does the representation *mean* (maps the sentences to the world)
- For example:
  color(my_car, red) → *??*
  '*my car is red*', '*paint my car red*', etc.

Inferential
- The interpreter
- Decides what kind of conclusions can be drawn
- For example: *Modus ponens (P, P→Q, therefore Q)*

Well-defined syntax/semantics

Knowledge representation languages should have precise syntax and semantics.

You must know exactly what an expression means in terms of objects in the real world.



Declarative vs. Procedural

Declarative knowledge *(facts about the world)*

- A set of declarations or statements.
- All facts stated in a knowledge base fall into this category of knowledge.
- In a sense, declarative knowledge tells us what a problem (or problem domain) is all about
- For example, Prolog is a declarative language

Procedural knowledge *(how something is done)*

- Something that is not stated but which provides a mean of dynamically (usually at run-time) arriving at new facts.

Declarative example

Information about items in a store

```
cheaper(coca_cola, pepsi)
tastier(coca_cola, pepsi)

if (cheaper(x,y) && (tastier(x,y) ) → buy(x)
```

Procedural example

Shopping script:

- Make a list of all items to buy
- Walk to the shop
- For each item on the list, get the item and add it to the shopping basket
- Walk to the checkout counter
- Pack the items
- Pay
- Walk home

Types of knowledge

Domain knowledge:

- What we reason about
- Structural knowledge
  - Organization of concepts
- Relational knowledge
  - How concepts relate

Strategic knowledge:

- How we reason
- At representation level, rather than at implementation level
  - (e.g. at implementation level – control knowledge, for resolving conflicting situations)

What is a Knowledge Representation?

"What is a Knowledge Representation?"
(Davis, Shrobe &Szolovits) *AI Magazine*, 14(1):17-33, 1993 http://groups.csail.mit.edu/medg/ftp/psz/k-rep.html

Defines the five roles the knowledge representation plays

- Role I: *A KR is a Surrogate*
- Role II: *A KR is a Set of Ontological Commitments*
- Role III: *A KR is a Fragmentary Theory of Intelligent Reasoning*
- Role IV: *A KR is a Medium for Efficient Computation*
- Role V: *A KR is a Medium of Human Expression*

Each role defines characteristics a KR should have

These roles provide a framework for comparison and evaluating KRs

Consequences of this KR

The spirit should be *indulged*, not *overcome* - KRs should be used only in ways that they are intended to be used, that is the source of their power.

Representation and reasoning are intertwined, i.e. a recommended method of inference is needed to make sense of a set of facts.

Some researchers claim equivalence between KRs, i.e. "frames are just a new syntax for first-order logic". However, such claims ignore the important ontological commitments and computational properties of a representation.

All five roles of a KR matter

Requirements for KR languages

Representation adequacy
- should to allow for representing all the required knowledge

Inferential adequacy
- should allow inferring new knowledge

Inferential efficiency
- inferences should be efficient

Clear syntax and semantics
- unambiguous and well-defined syntax and semantics

Naturalness
- easy to read and use

Knowledge representation formalisms

Production systems, expert systems

Semantic networks

Frames

Biologically inspired approaches

- neural networks
- genetic algorithms

Case-based reasoning

…

Semantic networks - history

Network notations are almost as old as logic

- Porphyry (3rd century AD) – tree-based hierarchies to describe Aristotle's categories
- Frege (1879) - *concept writing, a* tree notation for the first complete version of first-order logic
- Charles Peirce (1897) – *existential graphs*

First implementation of semantic networks

- Masterman (1961) – defines 100 concept types which are used to define 15,000 concepts organised in a lattice
- Quillian (1967) – *word concepts*, defines English words in terms of other words

Semantic networks - history

Developed by Ross Quillian, 1968, as "*a psychological model of associative memory*".

Associationist theories define the meaning of an object in terms of a network of associations with other objects in a domain or a knowledge base.

Quillian's model of a semantic network is based, not only on this idea of networks of information and knowledge, but on evidence that we also organise that knowledge hierarchically.

Semantic networks - history

The structure of their networks was devised from laboratory testing of human response times to questions.

They were asked questions such as “Is a canary a bird?”, “Can a canary sing?”, or “Can a canary fly?”

Quillian’s tests showed that response times for questions such as “Can a canary fly?” were longer than “Can a canary sing?”.

The results of this test were analysed and Quillian concluded that humans store information at its most abstract level

Semantic networks - history

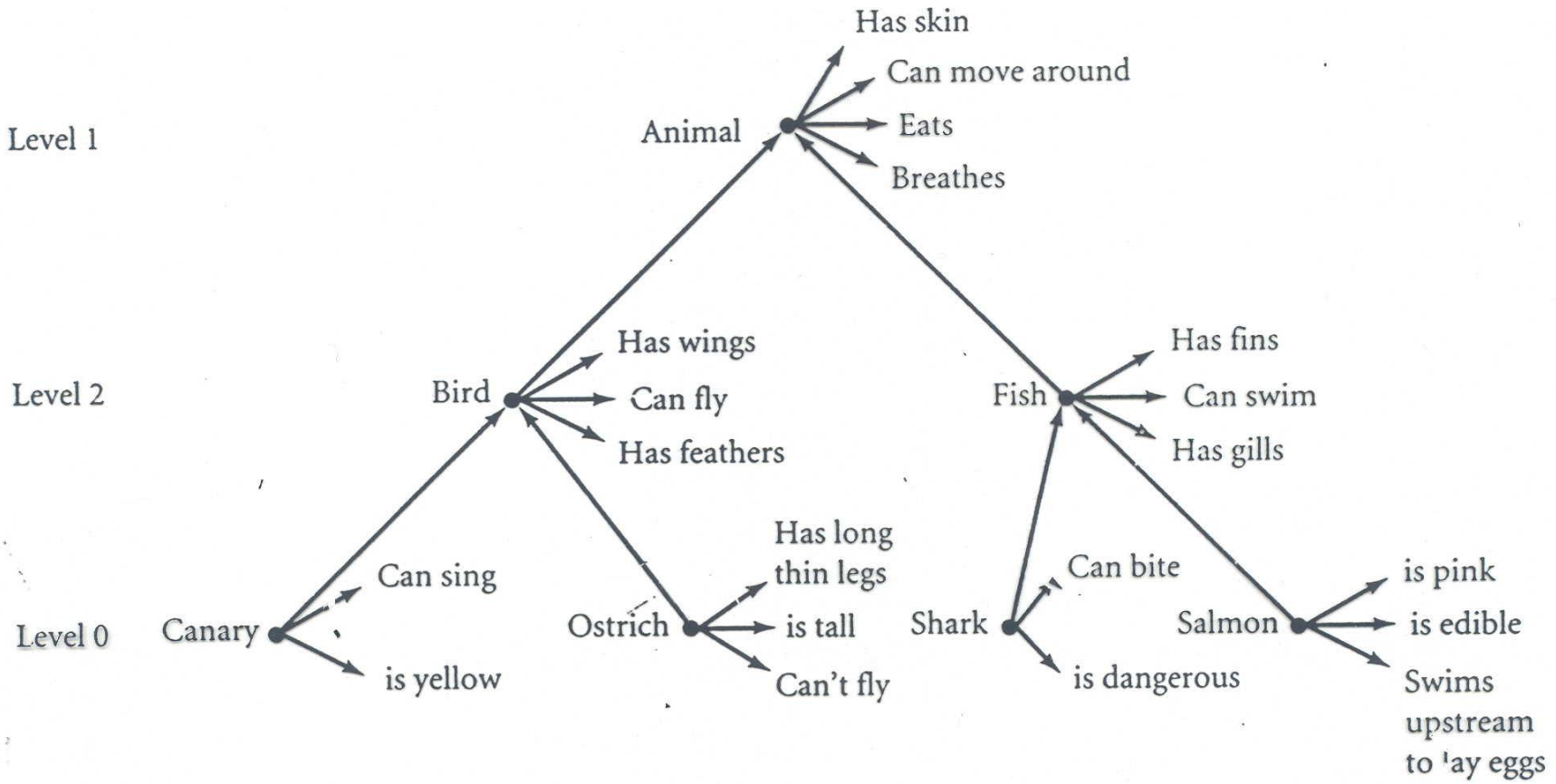


FIGURE 5.8: Hypothetical memory structure for a three-level hierarchy. (Adapted from Collins and Quillian, 1969. Reprinted by permission of Academic Press.)

Semantic networks

Instead of trying to recall that a canary flies, and a robin flies, and a swallow flies etc., humans remember that canaries, robins, swallows etc. are birds and that birds usually have an associated property of flying.

More general properties, such as eating, breathing, moving etc. are stored at an even higher level associated with the concept *animal*.

Thus reaction times to questions such as "Can a canary breathe?" were even longer again as more travel up the hierarchy is necessary to determine the answer.

The fasted recall was for traits specific to the bird such as "Is a canary yellow?" or "Can a canary sing?"

Handling exceptions

Handling exception cases also appears to be done at the most specific level.

Consider the properties associated with an *ostrich* in the semantic network. When subjects were asked if an ostrich could fly, the answer was produced faster than when applying the same question to canaries.

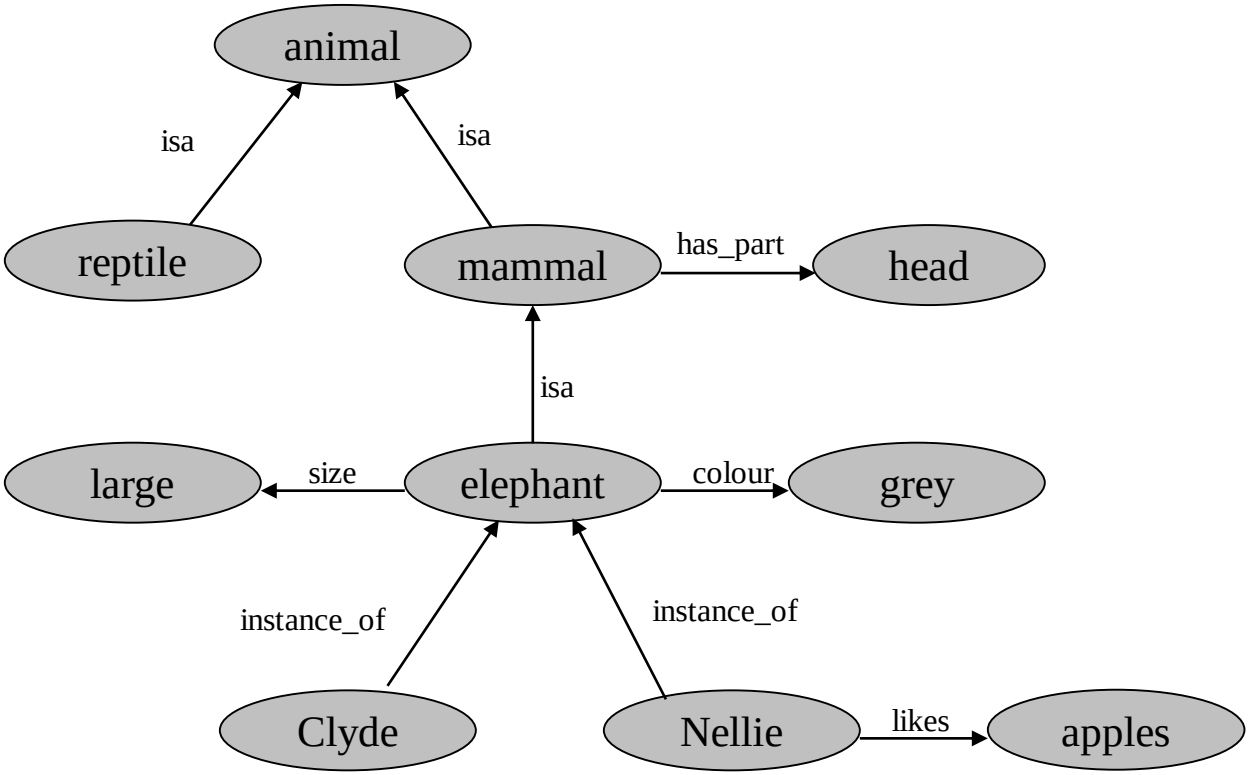
The conclusion reached by this is that the hierarchy

ostrich -> bird -> animal

is not traversed in order to understand this exception information.

Inheritance systems allow us to store information at the highest level of abstraction – this then reduces the size of the knowledge base and helps prevent update inconsistencies.

What is a semantic network?



Semantic networks: syntax

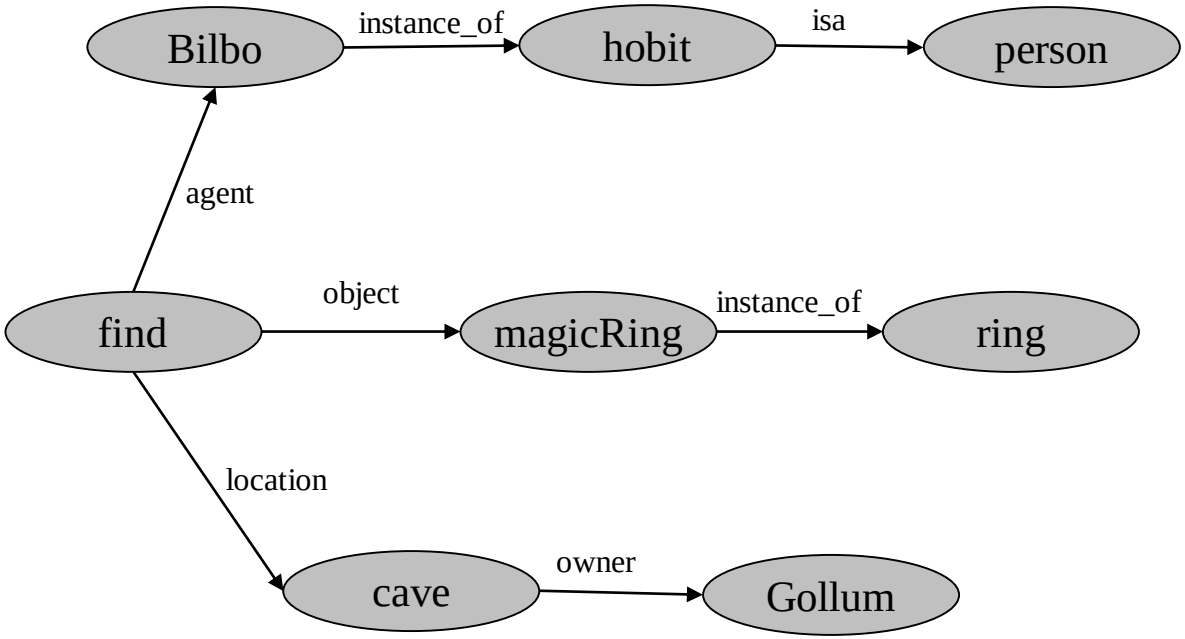
Represented as a **graph**.

Nodes represent **concepts**, **actions** or **objects** in the world.

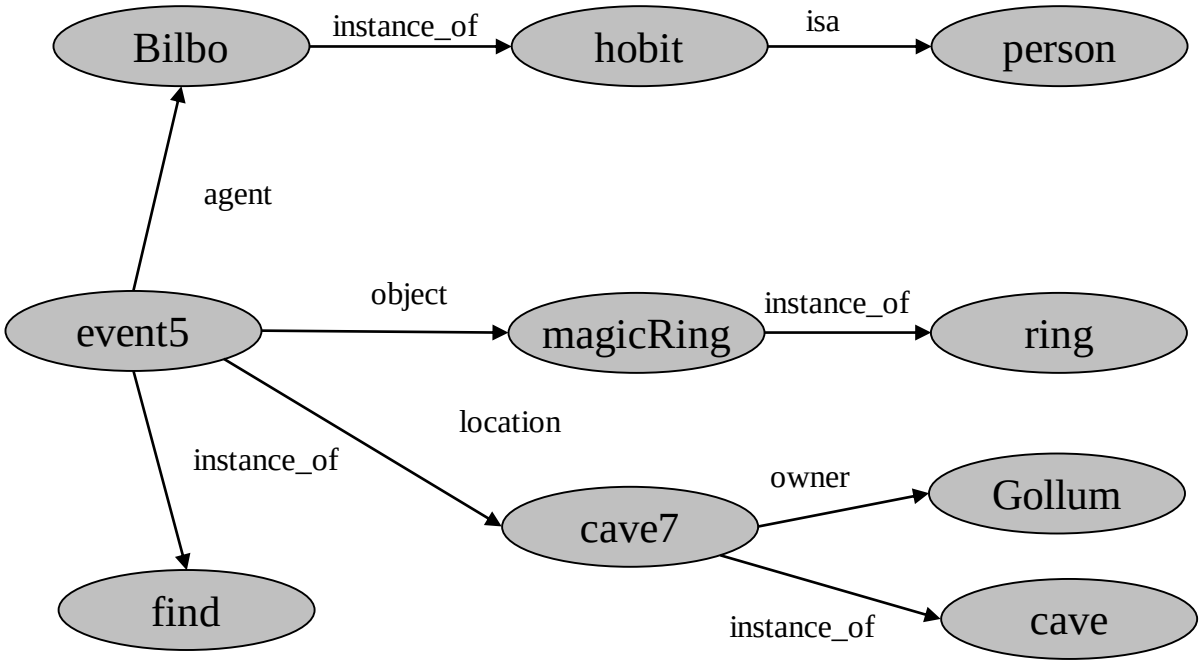
Links represent directional and labeled relationships between the nodes.

- Inheritance-oriented links: ”isa”, ”instance_of”
- General links: ”has_part”, ”causes”
- Domain-specific links: ”has_disease”, ”father_of”

Semantic network – example 1



Semantic network – example 2



Problems with semantic networks

Naming standards

- lack of naming standards for relationships
- naming of nodes

  If a node is labelled 'chair' does it represent:

  - A specific chair,
  - The class of all chairs,
  - The concept of a chair,
  - The person who is the chair of a meeting?

For a semantic network to represent ***definitive knowledge***, i.e. knowledge that can be defined, the relationship and node names must be rigorously standardised.

Often it is hard to distinguish between

- statements about object relationships with the world, and
- properties of the object

Problems with semantic networks

Searching

- Possible combinatorial explosion, especially if the response to a query is negative.
- Semantic networks, as we know, were originally devised as models of human associative memory in which one node has links to others and information retrieval occurs due to a spreading activation of the nodes.
- However, other forms of reasoning must be available to the brain, as it does not take a long time to answer the question "Is there a football team on Pluto?"

Unable to represent

- Negation
- Quantification
- Disjunction

Frames

Frames are a variant of semantic networks

A frame is

> *"a data structure for representing a stereotypical situation like …going to a child's birthday party"* (Minksy 1981).

All the information relevant to a concept is stored in a single complex entity called a **frame**.

Superficially, frames look like record data structures or class, however, at the very least frames also support inheritance.

```
class  Book {
        Person author;
        String title;
        int  price;
}
```

Frames

In frame-based systems we refer to

- **objects** – *Mammal*, *Elephant*, and *Nellie;*
- **slots** – properties such as *colour* and *size;*
- **slot-values** – values stored in the slots, e.g. *grey* and *large*.

Slots and the corresponding slot-values are inherited through the class hierarchy

Frames - example

The previous example can be also represented as frames:

Mammal:

subclass:	Animal
has-part:	head

Elephant:

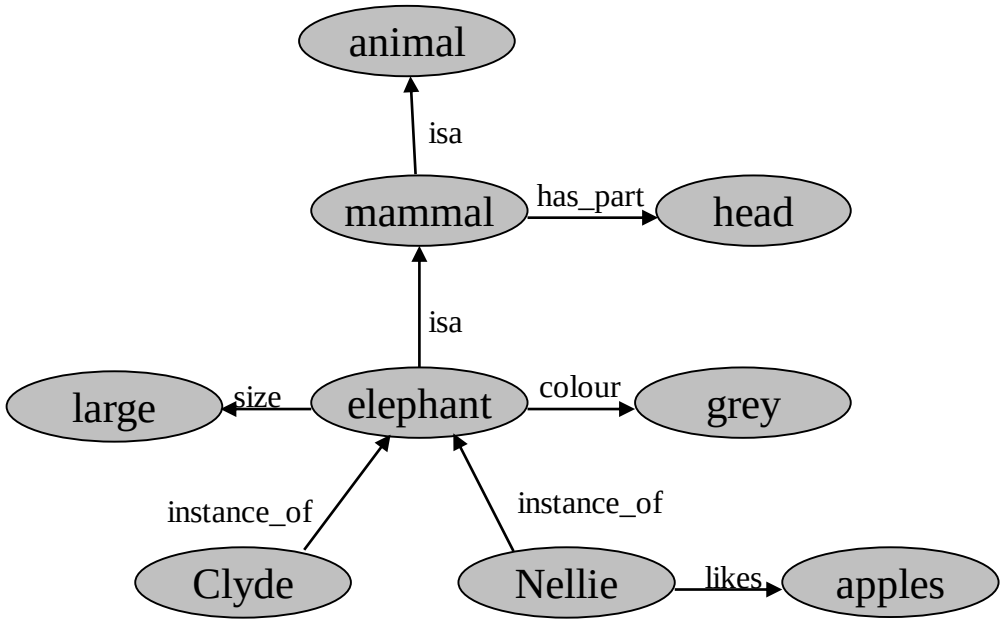
subclass:	Mammal
colour:	grey
size:	large

Nellie:

instance:	Elephant
likes:	apples

Clyde:

instance:	Elephant



Inheritance and default values

In general children classes inherit the properties of the parent class

Default values - properties that are typical for a class

Instances or subclasses whose properties differ from these default values are able to override them.

There are various ways of achieving overriding, for example:

- any default value may be overridden.
- only marked slots allow the default value to be overridden.

Default values

In this example only slots marked with an asterisk may be overridden.

Mammal:		
	subclass:	Animal
	has-part:	head
	warm-blooded:	yes
	*furry:	yes
Elephant:		
	subclass:	Mammal
	*colour:	grey
	*size:	large
	*furry:	no
Nellie:		
	instance:	Elephant
	likes:	apples
	owner:	Fred
	colour:	pink
Clyde:		
	instance:	Elephant
	size:	small

Inheritance

Children classes inherit the default properties of their parent classes unless they have an individual property value that conflicts with the inherited one.

We use the term simple (or single) inheritance when each object and class has only a single parent class.

Multiple Inheritance considers those cases where there is more than one parent class (e.g. Clyde is an instance of both Elephant and Circus-Animal)

- The frame system must be able to decide on precedence of inheritance

The complication occurs when some property may be inherited from more than one parent class. Some kind of mechanism is required to select which class the property is to be inherited from.

- The simplest solution is to define an order of precedence for the parent classes.

Multiple Inheritance

Let us consider the Elephant example where we consider a Circus-Animal:

Elephant:			Cylde:		
	subclass:	Mammal		instance:	Circus-Animal Elephant
	has-trunk:	yes		colour:	pink
	*colour:	grey		owner:	Fred
	*size:	large			
	*habitat:	jungle	Nellie:		
				instance:	Circus-Animal
Circus-Animal:					
	subclass:	Animal			
	habitat:	tent			
	skills:	balancing-on-ball			
	*size:	small			

What about the size of Clyde? How can we inherit 'habitat' from Circus-Animal but 'size' from Elephant?

Slots and procedures

Both slot values and slots may be frames.

In the multiple inheritance example, we represented that Fred was Clyde's owner. We may want to know some details about Fred, so we can use a frame to describe Fred's properties.

Allowing a slot to be a frame means that we can specify a range of properties for a slot.

- For example, we could specify that the slot **owner**
  - could only take the values of the class **Person**,
  - has an inverse slot **owns,** and
  - can take multiple values (i.e. a person can own many things).

Slots and procedures

Many systems allow slots to include **procedures**. The term, **procedural attachment** is used to represent this.

A piece of program code may be placed in a slot and be run every time a value for that slot is needed.

The code may also be run when a value is entered into the slot (event driven code). This code could do consistency checks or be used to propagate results to other slots.

The use of procedures and multiple inheritance can make it hard to predict what will be inferred about a given object just by looking at the set of frames. We need to know about the underlying system (the inference engine).

We say that the system has a **procedural**, rather than a **declarative semantics**, as the precise meaning of the frames depends on how the system infers new knowledge.

Inference in semantic networks and frames

Semantic networks and frames provide a fairly simple and clear way of representing the properties and categories of objects.

A basic type of inference Is defined whereby objects may inherit properties of parent objects.

However, inheriting properties from more than one parent, or defining conflicting properties if often problematic.

Questions?